

pg



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/746,949	12/21/2000	David M. Gillies	50037.04US01	2378

23552 7590 04/21/2004

MERCHANT & GOULD PC  
P.O. BOX 2903  
MINNEAPOLIS, MN 55402-0903

EXAMINER
----------

VO, TED T

ART UNIT	PAPER NUMBER
----------	--------------

2122

DATE MAILED: 04/21/2004

7

Please find below and/or attached an Office communication concerning this application or proceeding.

# Office Action Summary

Application No.

09/746,949

Applicant(s)

GILLIES ET AL.

P24

Examiner

Ted T. Vo

Art Unit

2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 27 January 2004.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-17 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-17 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

**DETAILED ACTION**

1. This action is in response to the amendment filed on 1/27/04.  
Claims 1, 8, 14, are amended.  
Claims 1-17 remain pending in the application.

***Response to Arguments***

2. Responsive to the previous office action, mailed date, 11/5/03, Applicants amended Claims 1, 8, and 14 presented in this office action, where Claims 1-3, 5, 8-10, 14-15 are rejected under 35 U.S.C. 102(b) as being anticipated by Tamches et al., "*Fine-grained Dynamic Instrumentation of Commodity Operating System Kernels*", Claims 4, 6-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamches et al., in view of Kiyohara et al., "Register Connection: A New Approach to Adding Registers into Instruction Set Architectures", and Claims 11-13, 16-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamches et al in view of Aho et al., "Compilers Principles, Techniques, and Tools".

Applicants' amendment of Claims 1, 8, and 14, necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicants' argument to the amended independent Claims 1, 8, and 14, over Tamches reference has been fully considered. However, it is not persuasive. For example, in their Remarks (page 5) Applicants argue that Tamches does not teach the limitation recited the amended Claim 1, including "*determining a maximum number of register requested from a plurality of register requests after the binary is compiled*", where Applicants contend that the teaching (Tamches) of dynamic instrumentation steps including live register analysis and patch location to hold generated code, and of Live register analysis in involving in determining registers that available for scratch use at instrumentation point, and of the

Art Unit: 2122

assumption of a maximum number of instructions needed to perform jump back (re: Remarks, page 5, third paragraph) are different from the limitation of Claim 1.

Examiner disagrees: Tamches reference discloses the limitation of Claims 1, 4, and 8, including the newly added feature, *"determining a maximum number of register requested from a plurality of register requests after the computer-executable binary is compiled"*.

Tamches teaches the Fine-grain Dynamic Instrumentation that includes loading and instrumenting code at machine instruction level without any need to recompile or reboot (re: Tamches: page 117, right column, started from third paragraph to the end of the column). The Fine-grain Dynamic Instrumentation inserts code patches *allocated from the patch area heap* in the kernel's address space (re: Tamches: page 119, right column section 3.1), and the Kernel debuggers can be implemented using fine grain dynamic instrumentation, where Breakpoints can be inserted at any machine code instruction by splicing code (re: Tamches: page 118, left column, fifth paragraph, and see "Splicing code" in section 3.4). Thus, Tamches' teaching includes handling patch code at machine level without any need to recompile or reboot; and therefore, the code is handled after the binary is compiled.

In the passage in page 120, second paragraph, section 3.3, fourth paragraph, Tamches teaches dynamic instrumentation steps including live register analysis and patch location to hold generated code, and of Live register analysis in involving in determining registers that available for scratch use at instrumentation point. These basis features are known as being participated in finding a number of free registers for register allocation. This is further indicated in Figure 4, page 121. *"1. Finding free registers before and after instrumentation point"*, and right in the same page 121, left column, second paragraph, *"The size of the machine code being inserted, extra instructions to spill some registers to the stack"*. It is known that the *spill* code ("a machine code") occurs when a determined number of registers in register allocation exceeds a maximum number of registers being used. If instructions (registers) excess a maximum number, some code must be spilled; otherwise, there would not be enough registers for allocation. Thus, through these passages, Tamches discloses **"determining a maximum number of register requested from a plurality of register"** as recited in Claim 1 and Claims 8, 14. Furthermore, within a passage in page 122, right column, first paragraph, Tamches discusses about a set of free

Art Unit: 2122

registers used for register scratching; these registers, and including the discussions as noted above, account for a **maximum number of register requested from a plurality of register.**

Thus, Applicants' argument given to the newly amended limitation of Claims 1, 8, and 14 in their Remarks (re: Remarks, page 5, third paragraph (lines 13-24)) under 35 U.S.C. 102(b) as being anticipated by Tamches is not persuasive as noted above.

***Claim Rejections - 35 USC § 102***

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1-3, 5, 8-10, 14-15 are rejected under 35 U.S.C. 102(b) as being anticipated by Tamches et al., "*Fine-grained Dynamic Instrumentation of Commodity Operating System Kernels*", February 1999, submitted in applicants' IDS.

Given the broadest reasonable interpretation of followed claims in light of the specification:

As per claim 1:

Tamches discloses a method that performs register allocation associated with a number of scratch registers. It uses a code patch, appeared as branch/jump instructions, and inserts at a predetermined location (instrumentation point) in a machine code instruction sequence (see page 119, first column, section 3, Mechanisms). Since code patch at each instrumentation point is jump or branch instructions, a needed maximum number of instructions is used to perform each instrumentation point (see page 121, first column, last paragraph). This implies to use a certain number of registers for very code patch.

Art Unit: 2122

The method uses live register analysis to determine registers that are available (*maximum number of registers*) for scratch use at an instrumentation point (see page 120, third and fourth paragraphs of section 3.3 Code Generation Issues). The method calculates the size of the patch used with scratch register allocation (see page 121, step 2 in figure 4, and second paragraph in the first column).

The teaching covers the claim limitations:

***"A computer-implemented method for obtaining scratch registers for use by a computer-executable binary, comprising:***

***(a) determining a maximum number of registers requested from a plurality of register requests***

(see Figure 4, page 121. "1. Finding free registers before and after instrumentation point" and page 121, left column, second paragraph, "The size of the machine code being inserted, extra instructions to spill some registers to the stack") ***after the computer-executable binary is compiled*** (re: Tamches: page 117, right column, started from third paragraph to the end of the column, page 119, right column section 3.1, and page 118, left column, fifth paragraph, "Kernel debuggers can be implemented using fine grain dynamic instrumentation, where Breakpoints can be inserted at any machine code instruction by splicing code [and see "Splicing code" in section 3.4], as referring to "from a plurality of register requests after the computer-executable binary is compiled); and

***(b) modifying each register request*** (see page 121, second paragraph of first column, 'the patch size is the sum: size of machine code being inserted, extra registers instructions to spill some registers to the stack') ***in the plurality of register requests to request the maximum number of registers plus an additional number of registers"*** (see page 121, second and third paragraphs of first column).

As per claim 2: Tamches discloses, "***The method of claim 1, wherein the additional number of registers corresponds to a selected number of scratch registers***" (see page 121, second paragraph of first column, 'extra registers instructions to spill some registers to the stack (if more scratch registers are needed than available)').

As per claim 3: Tamches discloses, "***The method of claim 1, wherein a procedure in the computer-executable binary includes the plurality of register requests***" (see page 121, second column, first

Art Unit: 2122

paragraph, 'the patch'; and page 122, figure 6, Code after Splicing; particularly '0x60060518 ba, a <patch addr>').

As per claim 5: Tamches discloses, "***The method of claim 1, further comprising (c) using at least one of the plurality of modified register requests to support instrumentation code in the computer-executable binary***" (see page 121, figure 4, and page 122, figure 5).

As per claim 8: Tamches discloses, "***A computer system, comprising:***

***(a) a computer-executable binary*** (see page 122; figure 6);

***(b) a procedure boundary detector configured to identify a procedure of the computer-executable binary*** (see Figure 4, page 121. "1. Finding free registers before and after instrumentation point" and page 121, left column, second paragraph, "The size of the machine code being inserted, extra instructions to spill some registers to the stack") ***after the computer-executable binary is compiled*** (re: Tamches: page 117, right column, started from third paragraph to the end of the column, page 119, right column section 3.1, and page 118, left column, fifth paragraph, "Kernel debuggers can be implemented using fine grain dynamic instrumentation, where Breakpoints can be inserted at any machine code instruction by splicing code [and see "Splicing code" in section 3.4], as referring to "from a plurality of register requests after the computer-executable binary is compiled); ***and***

***(c) a scratch register allocator configured to receive the identified procedure from the procedure boundary detector and to modify the computer-executable binary to request scratch registers*** (see page 121, figure 4, particularly, five steps in Instrumentation Step).

As per claim 9: Tamches discloses, "***The system of claim 8, wherein the computer-executable binary comprises at least one register allocation request***" (see page 122, figure 6, Code before Splicing; particularly '0x60060518 call 0x6015b818').

As per claim 10: Tamches discloses, "***The system of claim 9, wherein the scratch register allocator provides at least one scratch register by modifying the at least one register allocation request***" (see page 122, figure 6, Code after Splicing; particularly '0x60060518 ba, a <patch addr>').

As per claim 14: Tamches discloses, "**A computer-readable medium having computer-executable instructions**" (Page 121, figure 4, "cost" column: Examiner note: This shows there are microinstructions performing 5 steps; the performance of each step is calculated in microseconds) **comprising:**

**(a) discovering a procedure in a computer-executable instructions;** (see Figure 4, page 121. "1. Finding free registers before and after instrumentation point" and page 121, left column, second paragraph, "The size of the machine code being inserted, extra instructions to spill some registers to the stack") **after the computer-executable binary is compiled** (re: Tamches: page 117, right column, started from third paragraph to the end of the column, page 119, right column section 3.1, and page 118, left column, fifth paragraph, "Kernel debuggers can be implemented using fine grain dynamic instrumentation, where Breakpoints can be inserted at any machine code instruction by splicing code [and see "Splicing code" in section 3.4], as referring to "from a plurality of register requests after the computer-executable binary is compiled); **and**

**(b) if a register allocation does not exist at the beginning of the procedure, inserting a register allocation**", (see figure 5, 1-5 steps).

As per claim 15: Tamches discloses, "**The method of claim 14, further comprising:**

**(c) determining a maximum number of registers requested in the procedure;** (see page 120, second paragraph, section 3.3, fourth paragraph, 'determines registers that are available for scratch use at the instrumentation point', and see page 121, first column, last paragraph, particularly, 'assuming the maximum number of instructions needed to perform a jump' [examiner note: maximum number of instructions corresponds to the maximum number of registers]); **and**

**(d) modifying each register request in the procedure to request the maximum number of registers requested plus a number of scratch registers"** (see page 121, second paragraph of first column, 'the patch size is the sum: size of machine code being inserted, extra registers instructions to spill some registers to the stack').



**Claim Rejections - 35 USC § 103**

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

A person shall be entitled to a patent unless –

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6. Claims 4, 6-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamches et al., "Fine-grained Dynamic Instrumentation of Commodity Operating System Kernels", February 1999, as submitted in applicants' IDS, and in view of Kiyohara et al., "Register Connection: A New Approach to Adding Registers into Instruction Set Architectures", IEEE, 1993.

Given the broadest reasonable interpretation of followed claims in light of the specification:

As per claim 4:

Tamches disclose, "**wherein the additional number of registers corresponds to a selected number of scratch registers**" (page 120, second column, fourth paragraph of section 3.3, 'registers that available for scratch use')

Tamches does not address, "**wherein each of the scratch registers is indexed by an index that remains constant throughout the procedure**".

Kiyohara, discloses a connection of a set of extended registers (scratch registers) (Kiyohara: page 248, figure 1, the extended registers are indexed from m, m+1,... n in register file). The extended registers are required in changes in register allocation process (Kiyohara: page 251, second column, first paragraph), where an existing set of register resource cannot be accommodated (Kiyohara: page 247, second column, lines 14-19). Allocation requires indexing to accommodate with the extension (Kiyohara: page

Art Unit: 2122

247, second column, line 38). Kiyohara's teaching covers the limitation, "**wherein each of the scratch registers is indexed by an index that remains constant throughout the procedure**".

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include the teaching of register indexing in accommodation with extended set of register in register allocation as disclosed by Kiyohara with the teaching using the scratch register in register allocation of Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming common principle of register allocation, where all the used set of registers must be indexing for allocation.

As per claim 6:

Tamches does not address, "**wherein the computer-executable binary is constructed for execution on a processor configured to execute a speculative instruction**".

Kiyohara, discloses scheduling techniques that re-order instructions (limitation: 'a processor configured to execute speculation instruction') (Kiyohara: see page 247, second column, first paragraph, "the code scheduling techniques reorder instructions"). Kiyohara's teaching covers limitation: "**wherein each of the scratch registers is indexed by an index that remains constant throughout the procedure**".

Speculative execution is known as providing application's subsequent performance. According to definition of Speculative execution, it is a technique that allows a superscalar processor to keep its functional units as busy as possible by executing instructions before it is known that they will be needed (appeared in internet: FOLDOC, on-line dictionary of computing). Functional units are known as logic units, memory address register, register file, etc.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to exploit instruction-level parallelism using speculation execution, as included in Kiyohara (re-order instructions), into the teaching of register scratching, as disclosed by Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming a common technique in instruction executions for application's subsequent performance.

As per claim 7:

Tamches does not address limitation of claim 7.

Kiyohara further discloses, "***The method of claim 6, wherein data is stored in a register in association with the speculative instruction, and wherein moving the data to a main memory results in a hardware fault.***" (Kiyohara: page 247, second column, first paragraph, "the code scheduling techniques reorder instructions so that instructions that are close to each other tend to be independent of each other"). Re-order instruction is a speculation technique that causes an instruction to move ahead. For example, an instruction is moved ahead a branch instruction. Moved instructions, thus, might cause an exception that results in a hardware fault.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to exploit instruction-level parallelism using speculation execution, as included in Kiyohara, into the teaching of scratch register allocation, as disclosed by Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming a common technique in instruction executions for application's subsequent performance, where the speculation might cause an exception to result in hardware fault.

7. Claims 11, 16-17 is rejected under 35 U.S.C. 103(a) as being unpatentable over Tamches et al., "*Fine-grained Dynamic Instrumentation of Commodity Operating System Kernels*", February 1999, as submitted in applicants' IDS, in view of Aho et al., "Compilers Principles, Techniques, and Tools", 1986.

As per claim 11: Tamches discloses, "***The system of claim 8, further comprising:***

***(d) a basic block detector configured to receive the identified procedure from the procedure boundary detector and to identify at least one basic block in the identified procedure;*** (Tamches: see page 120, first column, section 3.2, particularly, third paragraph of section 3.2);

Regarding further limitation:

***(e) a dominating register allocation detector configured to receive the at least one basic block and to detect at least one dominating allocation for the at least one basic block wherein the scratch***

Art Unit: 2122

***register allocator is further configured to receive the at least one basic block identified and the at least one dominating allocation detected:***

Tamches discloses the code generation that configures to perform scratch register allocation to assign scratch registers at an instrumentation point (Tamches: see page 121, figure 4; particularly, steps 1-4). The allocation is performed after a structure analysis (Tamches: particularly third paragraph of section 3.2) which analyzes live registers (a formal requirement for identifying a set of scratch register used at the instrumentation point). Tamches concerns splicing for hazard-free when it addresses a single instruction at instrumentation point (Tamches: page 123, section 3.4.2; see whole contents in first column); and it concerns about un-safety when it addresses multiple-instruction splicing, involving a plurality of basis blocks (Tamches: page 123, section 3.4.2; see whole contents in second column). The disclosure covers a part (dominating register) of limitation (e).

Tamches does not address, "dominating register allocation", where "***the scratch register allocator is further configured to receive the at least one basic block identified and the at least one dominating allocation detected***".

Aho teaches means for forming one basic block and finding dominators at graph analysis phases (Aho: see page 670-671). Aho shows a reduction of a flow graph might be transformed in flow graph analysis (Aho: page 668, example 10.37). Aho teaches finding Dominators in the flow graph consisting a plurality of nodes (basis block). The finding dominators assists identifying some regions in a flow graph might be applied for transformation (Aho: page 669, four paragraph, 'If we instead use T2...'). Thus finding dominators is necessary for optimization in speeding up flow analysis and reducing register usage. Aho's teaching covers limitation: "dominating register allocation" and "dominating allocation detected".

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include a necessary step, "finding dominators", as taught by Aho, into the flow analysis before performing scratch register allocation at an instrumentation point as discloses by Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming to a requirement of optimization technique in flow graph analysis for reducing a number of register usages due to scratching involvement.

As per claim 16: Tamches addresses each basis block in a procedure is analyzed for finding live registers. Tamches uses live registers in an association with scratch register allocation (see page 120, section 3.2 Structure analysis, and see figure 5, 1-5 steps).

Tamches does not explicitly address:

***(c) for a basic block in the procedure (i) finding at least one dominating allocation; (ii) modifying the at least one dominating allocation to request a number of scratch registers.***

Aho teaches means for forming one basic block and finding dominators at graph analysis phases (Aho: see page 670-671). Aho shows a reduction of a flow graph might be transformed in flow graph analysis (Aho: page 668, example 10.37). Aho teaches finding Dominators in the flow graph consisting a plurality of nodes (basis block). The finding dominators assists identifying some regions in a flow graph might be applied for transformation (Aho: page 669, fourth paragraph, 'If we instead use T2...'). Thus finding dominators is necessary for optimization in speeding up flow analysis and reducing register usage. Aho's teaching covers limitation: ***"(i) finding at least one dominating allocation; (ii) modifying the at least one dominating allocation to request a number of scratch registers"***.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include, "finding dominators", as taught by Aho, into the flow graph analysis before performing scratch register allocation at an instrumentation point as discloses by Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming to a requirement of optimization technique in flow graph analysis for reducing register usages due to scratching involvement.

As per claim 17: Tamches addresses each basis block in a procedure is analyzed for finding live registers in association with scratch register allocation (see page 120, section 3.2 Structure analysis, and

see figure 5, 1-5 steps) to cover claim limitation: ***"(c) finding at least one basic block in the procedure; (d) constructing a control flow graph from the at least one basic block;***

Tamches does not explicitly address: ***"(e) using the control flow graph to discover at least one dominating allocation; and (f) modifying the at least one dominating allocation to request a number of scratch registers.***

Aho teaches means for finding dominators at flow graph analysis phases (Aho: see page 670-671). Aho shows a reduction of flow graphs through transformations (Aho: page 668, example 10.37). Aho's teaching of finding Dominators is used for code generation that employs register allocation (Aho: page 587, figure 10.1, 'code generation' → target code' - - - compiler can use registers selections do peephole transformations). Aho's teaching covers limitation: ***"(e) using the control flow graph to discover at least one dominating allocation; and (f) modifying the at least one dominating allocation to request a number of scratch registers".***

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include, "finding dominators", as taught by Aho, in to the flow graph analysis before performing scratch register allocation at an instrumentation point as disclosed by Tamches. The modification would be obvious because one of ordinary skill in the art would be motivated for conforming to a requirement of optimization technique in flow graph analysis for register allocation.

8. Claims 12-13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamches et al., "Fine-grained Dynamic Instrumentation of Commodity Operating System Kernels", February 1999, as submitted in applicants' IDS, in view of Aho et al., "Compilers Principles, Techniques, and Tools", 1986, and in further view of Tamches.

As per claim 12: Claim 12 is dependent on claim 11. However, the claim limitation, ***"The system of claim 11, wherein the basic block detector is further configured to construct a control flow graph using***

Art Unit: 2122

***the at least one basic block identified*** is further limitation of step (d) in the claim 11, where the teaching of step (d) is identified by Tamches' as addressed and set forth in the issue of claim 11 rejection.

Tamches further discloses the Kerninstd (examiner interprets as basis block detector) that performs partitioning and identifying a set of live registers for use in register allocation (Tamches: see page 120, first column, section 3.2; particularly, see second and third paragraphs of this section). This teaching covers the claim limitation, ***"The system of claim 11, wherein the basic block detector is further configured to construct a control flow graph using the at least one basic block identified"***.

As per claim 13: Claim 13 is dependent on claim 12. Where Tamches' teaching of claim 12 is identified and addressed in the issues of Claim 12 rejection above.

Tamches further discloses, ***"The system of claim 12, wherein the scratch register allocator is further configured to receive the control flow graph"*** in figure 4.

(See figure 4, step 1: Finding free register before and after instrumentation point. Examiner interprets that "Finding free registers" has means of configuring scratch register allocation; "before and after" has means of receiving information from control flow graph analysis: (see page 120, section 3.2, Structure Analysis, for covering control flow graph)).

With regards to further teaching of Tamches to claims 12-13:

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to include a necessary step, "finding dominators", as taught by Aho, into configuring flow analysis as disclosed by Tamches.

The modification would be obvious because one of ordinary skill in the art would be motivated for conforming to a requirement of optimization technique in flow graph analysis for reducing number of register usages in scratch register allocation.

**Conclusion**

9. Applicant's amendment to newly added limitations in Claims 1, 8 and 14 necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (703) 308-9049. The examiner can normally be reached on Monday-Friday from 8:00 AM to 5:30 PM ET. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam, can be reached on (703) 305-4552.

The fax phone numbers:

(703) 872-9306 (for formal communication intended for entry);

(703) 746-5429 (for informal or draft communication, please label "PROPOSED" or "DRAFT").

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-3900.

TTV  
Art Unit: 2122  
April 19, 2004

  
**TUAN DAM**  
**SUPERVISORY PATENT EXAMINER**